Frågor: - Ska man köra med -Pn flaggan på nmap? Det kan väl ta galet lång tid då.

Scanning

Configure your setup - route external machine

If it is the case that you have for example Nessus or some other tool on an external machine, but you do not want to connect it straight into your network, you can route the nessus machine through your pentest machine that is connected to the network.

What you will basically do here is make your pentest machine into a router. And then connect your external machine to it, and route the traffic through through your router.

1. Connect external machine via ethernet to your pentest machine.
2. Give external machine ip.

```
ip a add 192.168.150.2/24 dev eth0
```

Bring the interface up again

```
ip link set eth0 up
```

3. Add routing rules

```
ip r add 10.0.0.0/16 via 192.168.150.1 dev eth0
```

4. Configure your MITM/forwarding machine

```
ip a add 192.168.150.1/24 dev eth1
```

```
iptables -t nat -A POSTROUTING -o tun0 -j MASQUERADE
echo 1 >  /proc/sys/net/ipv4/ip_forward
```

Using MSF

One good idea is to scan the network in metasploit. This will allow you to search its database. This will be necessary if the network is big.

https://www.offensive-security.com/metasploit-unleashed/using-databases/

```
msfconsole
db_nmap



## List hosts and ports
services


## List hosts
hosts


## Import nmap file
db_import /path/to/file


## Export to CSV file
services -o file.csv
```

Scan safely

Nmap should not crash applications. It does not send malformed packets. If an application crashes it is because it is badly written. If they do crash the vendor should be alerted and the problem fixed.

- Use SYN scan (-sS) instead of connect scan (-sT). User-mode applications such as web servers can rarely even detect the former because it is all handled in kernel space and thus the services have no excuse to crash.
- Version scanning (-sV) and some of our NSE scripts (-sC or --script) risk crashing poorly written applications. Similarly, some buggy operating systems have been reported to crash when OS fingerprinted (-O). Omit these options for particularly sensitive environments or where you do not need the results.
- Using -T2 or slower (-T1, -T0) timing modes can reduce the chances that a port scan will harm a system, though they slow your scan dramatically. Older Linux boxes had an identd daemon that would block services temporarily if they were accessed too frequently. This could happen in a port scan, as well as during legitimate high-load situations. Slower timing might help here. These slow timing modes should only be used as a last resort because they can slow scans by an order of magnitude or more.
- Limit the number of ports and machines scanned to the fewest that are required. Every machine scanned has a minuscule chance of crashing, and so cutting the number of machines down improves your odds. Reducing the number of ports scanned reduces the risks to end hosts as well as network devices. Many NAT/firewall devices keep a state entry for every port probe. Most of them expire old entries when the table fills up, but occasional (pathetic) implementations crash instead. Reducing the ports and hosts scanned reduces the number of state entries and thus might help those fragile and defective devices stay up.

Discover network

The first thing you need to do is to get to know the network a bit.

I usually try to inspect the network traffic a little bit before doing anything. NOTICE, that you will perform a DORA (dicovery, offer, request, ack) over DHCP. Iptables will thus NOT block dhcp-requests. I configure iptables to drop all traffic, to make sure I don't interact with it in any way.

```
 iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

ip6tables -P INPUT DROP
ip6tables -P OUTPUT DROP
ip6tables -P FORWARD DROP
```

NOTICE that when you run wireshark you will see your outgoing traffic even through iptables is droppping it. That is because wireshark is hooked up to the interface before iptables. So you will also see incoming traffic even though iptables drops it.

Just connect your machine to the ethernet or wifi, then check what ip-address and netmask you get.

```
 ip a
wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    inet 10.10.1.10/24 brd 10.10.1.255 scope global dynamic wlan0
```

In this example you get a netmask of /24 (255.255.255.0).

You can easily see the network range, broadcast address and how many hosts by using `sipcalc`

```
sipcalc eth0
```

Discover hosts

Discovering hosts is fundamental. It is not impossible that the network you are on is a /8. That would be 16 million addresses. So we can't send a SYN packet to each port on all those 16 million hosts. That would take forever. In NMAP host discovery actions are called PING SCANS, although they are much more than simple ICMP Pings.

The challenge in discovering hosts is that you don't have all the time in the world. So you need to balance between time and result. More time gives you better result (if you port scan each port on all machines for example). You might not have the possibility to scan everything, because the network might be too big. Then you need to first identify all hosts, and then scan those.

Disable jost discovery

You can disable host discovery, or ping scans with this flag

```
-Pn
```

If the flag is included nmap will send packets to each host and port no matter what. Independet of if it doesn't exist. This might sometimes be useful if the host is not responding to any of the ping scan techniques, but is in fact open and exposes a port.

If you want to be 100% sure that you are not missing any hosts this option might be useful.

Exclude hosts

If you have some really critical systems that you don't want to scan. ANd you don't want to get blamed if they go down, you can add an exclude-list

```
--exclude, --excludefile <filename>
```

Another good idea is simply to drop all packets to those IP:s from your iptables rules.

Remember to take a tcpdump of all the traffic, this way you can prove that it wasn't you.

Perform a dry run

To make sure that you haev added all the target flags correctly, you can always perform a dry run. Nmap will only make reverse DNS resolutions, but not packets are sent to the hosts. For example:

```
nmap -sL 10.0.0.1/25
```

Discover hosts

Okay, not lets get down to it.

1. Very stealthy - Perform a simple reverse DNS resolution

```
nmap -sL 10.0.0.1/24
```

This will make a DNS reverse resolution. This is an easy way to discover some hosts through the DNS.

2. Less stealthy - Ping sweep The `-sn` flag will disable the port scan. It perform a icmp ping, TCP SYN to 443 and TCP ACK to 80, and an ARP scan.

```
nmap -sn 10.0.0.1/24
```

This is actually a relly good, kind of stealthy command, but still useful.

It is equivalent to doing this:

```
nmap -PE -PS443 -PA80 -PP -PE 10.0.0.1/24
```

Basically, the possibilities you haev to discover a host are the following: - ARP broadcast - ICMP PING - TCP SYN to specified port (80/443) - TCP ACK to specified port (80/443) - UDP PING

THe difference an attacker can make i to change the TCP and UDP ports.

- IP Protocol PING

Discover services (ports)

Terminology

Nmap defines the state of a port according to the following:

open

```
An application is actively accepting TCP connections or UDP packets on this port. Finding these is often the primary goal of port sc
```

closed

```
A closed port is accessible (it receives and responds to Nmap probe packets), but there is no application listening on it. They can
```

filtered

```
Nmap cannot determine whether the port is open because packet filtering prevents its probes from reaching the port. The filtering co
```

unfiltered

```
The unfiltered state means that a port is accessible, but Nmap is unable to determine whether it is open or closed. Only the ACK sca
```

open|filtered

```
Nmap places ports in this state when it is unable to determine whether a port is open or filtered. This occurs for scan types in whi
```

closed|filtered

Port scanning

In order to decrease the traffic perform a host-discovery first. It will decrease the traffic but it will also increase the risk that

```
db_nmap -iL
```

Test network segmentation

Network segmentation is the act of splitting networks into different segmentation.

Segmenting the network can server different purposes, for example as a security measure to create strict walls between groups of servers/computers. Segmenting the network can limit the damage from an attack. If the attacker only access for example the client network, and not the servers. It will be more difficult for the attacker to reach critical data.

Reasons why network segmentation is a good idea.

- Improved Security. Network traffic can be isolated and / or filtered to limit and / or prevent access between network segments.
- Better Access Control. Allow users to only access specific network resources.
- Improved Monitoring. Provides an opportunity to log events, monitor allowed and denied internal connections, and detect suspicious behavior.
- Improved Performance. With fewer hosts per subnet, local traffic is minimized. Broadcast traffic can be isolated to the local subnet.
- Better Containment. When a network issue occurs, its effect is limited to the local subnet.

Test

The easy way to test network segmentation is simply to access one segment. Ask someone with knowledge of the network for a host and port on another network segment, and the simply try to connect to that host.

Try an ICMP connect SYN

Test for VLAN-hopping.

Vulnerability scanning

Most of the time you will want to do a vulnerability scan of the network. A vulnerability scan will be more intrusive and send more aggressive payloads, than a simple nmap scan. PORTS TO NOT SCAN: 9100, 515, 3396, 9303. These are ports used for printers, and can cause some kaoz. Taken from lsit here by searhing for printers: [https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers][Printer ports]

Nessus

Openvas

1. Create a target Click on "Configuration" and then on the little start (*) on the left. If you hover over it it says "New Target".

Configure the target. Add the range and other settings.

2. Configure scanning task

Go to Scans / Tasks. Then click on the little star to the left. If you hover over it is says "Create new task".

In scan config you can set it to Full and fast. It all depends on how much time you have.

Basic test - Non-intrusive - Passive - TCPDump

The easiest test is simply to run tcpdump or wireshark while you are on the network and observe the traffic.

Identify routes